# Livingstone Model-Based Diagnosis of Earth Observing One Infusion Experiment

Sandra. C. Hayden * Adam J. Sweet†
*QSS Group, Inc, Moffett Field,CA, 94035, USA*

Scott E. Christa‡
*AerospaceComputing, Inc., Moffett Field, CA, 94035, USA*

**The Earth Observing One satellite, launched in November 2000, is an active earth science observation platform. This paper reports on the progress of an infusion experiment in which the Livingstone 2 Model-Based Diagnostic engine is deployed on Earth Observing One, demonstrating the capability to monitor the nominal operation of the spacecraft under command of an on-board planner, and demonstrating on-board diagnosis of spacecraft failures. Design and development of the experiment, specification and validation of diagnostic scenarios, characterization of performance results and benefits of the model-based approach are presented.**

## I. Introduction

IN this experiment, Livingstone is uploaded to the Earth Observing One (EO-1) satellite to conduct diagnostic tests. EO-1 was developed and is operated by NASA Goddard Space Flight Center (GSFC). Livingstone is a diagnostic engine developed at NASA Ames Research Center (ARC) by the Model-Based Diagnosis and Recovery group. The original Livingstone flew on Deep Space 1 (DS1) as part of the Remote Agent autonomy experiment (RAX) in 1999 [2]. Since then, the group has created the next version of Livingstone, called Livingstone 2 (L2). As a technology infusion experiment, L2 and the spacecraft diagnostic models have been integrated with the Autonomous Sciencecraft Experiment (ASE) [3]. ASE was developed by NASA Jet Propulsion Laboratory (JPL), and first ran on-board the EO-1 on September 20, 2003. The autonomy software consists of the Continuous Activity Scheduling, Planning, Execution and Replanning (CASPER) planner; the science event detection software and the Spacecraft Command Language (SCL), developed by Interface and Control Systems (ICS). SCL provides an executive, a database and the software bridge to the spacecraft's 1773 data bus. L2 provides a diagnosis component to ASE, which was not included before.

A range of development and integration activities were undertaken to support the experiment. The major challenges encountered and their resolutions are discussed. Tasks include development of L2 models of the EO-1 spacecraft and instrumentation, and failure scenario definition, based on knowledge acquired from GSFC. A Real-Time Interface (RTI) and corresponding modeling methodology were developed to account for communication delays and physical transients in the system. L2, models of the spacecraft, and the RTI were integrated with SCL and the CASPER planner, first on a PowerPC embedded system and then on the EO-1's ground test bed, a pair of M-5 processors with the Virtual Satellite (VSat) simulation system. Diagnostic scenarios were validated prior to upload on integrated system. System engineering of the overall autonomy software included allocation of VxWorks task priorities, SCL software bus bandwidth, CPU and RAM resources. In early September 2004, the combined L2 and ASE software was uploaded to EO-1, and checkout procedures of the L2 software began.

* Group Manager, Computational Sciences Division, NASA Ames Research Center, M/S 269-3, and Professional Member.
† Research Engineer, Computational Sciences Division, NASA Ames Research Center, M/S 269-1, AIAA Professional Member.
‡ Systems Engineer, NASA Ames Research Center, M/S 247-2

## II. Livingstone Diagnostic Technology

The Livingstone algorithm and component-connection model are introduced here. A Livingstone diagnosis system consists of two main parts, a generalized inference engine and a domain-specific model. When Livingstone is deployed on different devices or vehicles, the inference engine does not change; only a new model needs to be developed. Livingstone uses a qualitative representation, propositional logic, to model the target system. The target system may be physical, such as the spacecraft hardware, or logical, such as the spacecraft software. The model is used to predict the states of system components given their initial state, commands which affect the system, and possible mode transitions. If there is a discrepancy between observed and predicted behavior, this generates conflicts in Livingstone's internal belief state. These conflicts are then used to focus the search for component modes (including failure modes), which are consistent with the observed state of the world and the possible mode transitions in the model. This process is known as *conflict-directed best-first search.* The set of component modes, which is found to satisfy the constraints expressed in the model, is termed the mode vector.

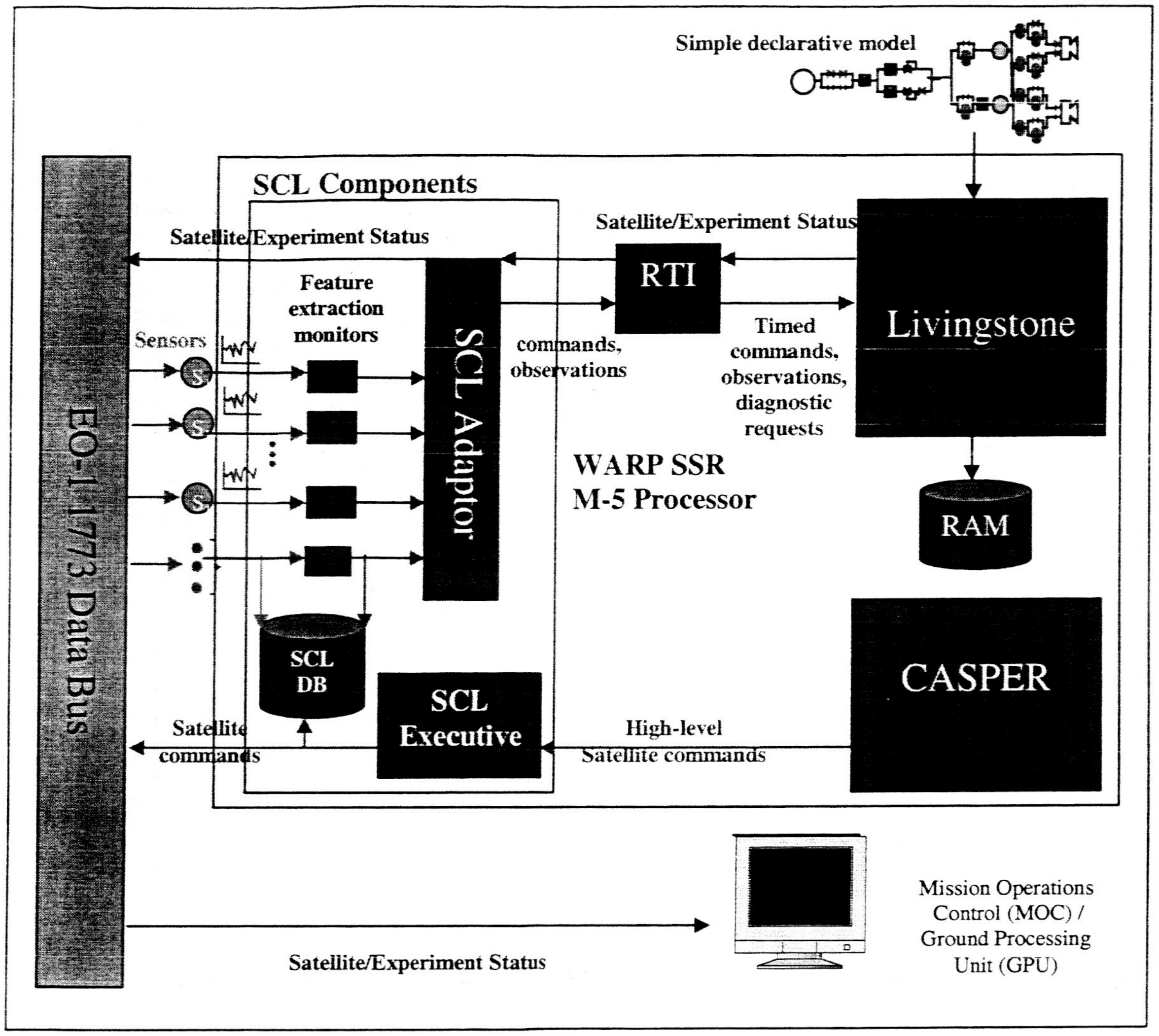
A Livingstone component-connection model describes nominal and failure modes for components in terms of the propositional constraints that must hold in those modes. The connections are the constraints that must hold due to interactions between the components. Transitions between modes of a component are triggered by guard conditions such as commands, in a finite state machine representation. Constraints are expressed as discrete variable-value pairs, giving rise to a qualitative model. The real-valued sensor data must be transformed into qualitative data ("binned") by software called monitors before being used by Livingstone. For the failure modes, the likelihood of the failure is indicated by a rank, an approximation of the prior probability. The mode vector describes the overall state of the system.

## III. Architecture of the Diagnostic Experiment

L2 was integrated with the Autonomous Sciencecraft Experiment software architecture and infrastructure, and uploaded to the WARP Mongoose-5 (M-5) processor on-board the EO-1 satellite. The experiment has the capability to process spacecraft telemetry and to downlink diagnostic health status telemetry for monitoring and display at the Mission Operations Center (MOC). The experiment architecture and configuration is shown in Figure 1.

Briefly, the CASPER planner generates high-level plan scripts and sends them to the SCL Executive that generates a sequence of commands to execute the plan. The SCL Software Bridge connects applications on the WARP-M5 processor to the 1773 spacecraft data bus for all telemetry processing, including decommutation of incoming telemetry frames and support for telemetry downlink. Incoming data may be stored in the Data Repository, with database triggers for notification of executed commands and received observations to subscriber processes such as L2.

L2 performs diagnosis using a qualitative model of the target system to predict observations given the commands issued to the system, postulating diagnoses to explain discrepant observations. A diagnosis consists of a group of failure candidates, their constituent modes and the likelihood of each candidate.



Simple declarative model
SCL Components
Satellite/Experiment Status
Satellite/Experiment Status
RTI
Livingstone
Feature extraction monitors
Sensors
SCL Adaptor
commands, observations
Timed commands, observations, diagnostic requests
WARP SSR M-5 Processor
RAM
EO-1 1773 Data Bus
SCL DB
SCL Executive
CASPER
Satellite commands
High-level Satellite commands
Mission Operations Control (MOC) / Ground Processing Unit (GPU)
Satellite/Experiment Status

**Figure 1: Architecture of the L2 on EO-1 Experiment.**

## IV. Livingstone Model Development

An L2 model contains the device- or vehicle-specific information used in diagnosis. L2 models are created in a component-based manner: first components are defined, then connected together to create the overall system model. Components can also be contained in other components in the model, although internally L2 treats the model as flat.

For speed considerations, L2 models are discrete. Variables can take a finite number of values such as "low", "medium", and "high", and components can contain a finite number of modes such as "on", "off", "failedOn", and "failedOff". In these respects, an L2 model resembles a finite-state machine. The input commands are used as the guards enabling the system to switch modes. Each component mode specifies qualitative constraints between the internal variables and the vehicle telemetry.

To perform diagnosis, L2 uses the commands given to the system to indicate component mode changes. Using the constraints in the model, L2 generates the expected telemetry values of the new system mode and compares them to the actual telemetry values. If the expected values and the actual values disagree, L2 determines that a fault has occurred. It then searches the space of component fault modes to find those consistent with the current observations; a diagnosis containing the set of consistent component faults is returned.

In general, the process of creating an L2 model has four steps: knowledge acquisition, scope definition, model creation, and model testing. In practice, these steps are often iterated before a model is complete.

## A. Knowledge Acquisition

Knowledge acquisition is the process by which a modeler gathers information about a device or vehicle. It is usually the most time-consuming part of creating an L2 model. It simply takes time to understand the components of a vehicle and how they behave in nominal and fault conditions. In addition, if a system is still in the design stage, the information to capture in a diagnosis model is in flux or may not exist. Usually several sources of information are used to gain the knowledge needed: design specifications, schematics and Failure Modes, Effects and Criticality Analysis (FMECA), and, if available, the system designers themselves.

For EO-1, the models were created by the group at Ames, supported by the GSFC engineers. The forms of documentation mentioned above were used, as well as mission timelines and the EO-1 Spacecraft User's Guide. EO-1 has been flying for several years, which makes knowledge acquisition easier: the satellite hardware is not changing, the documentation is more complete, and the engineers have years of experience operating the spacecraft.

## B. Scope Definition

Determining the scope of an L2 model involves deciding which vehicle components and component faults to include in the model, and the level of detail in which to model them. The scope of the EO-1 L2 model is a subset of the spacecraft components most relevant to the science data collection sequence: the two imaging instruments, called the Hyperion and the Advanced Land Imager (ALI), and the data recording device, called the WARP. To ease the integration of LEO-1 with ASE, the model scope was chosen to require only a subset of the commands and telemetry already used by ASE. The telemetry values in use by ASE are mostly discrete "status bit" values. As a result, the EO-1 model is fairly high-level. More detailed models could be developed with additional work and by incorporating additional telemetry values.

## C. Model Creation

After gaining knowledge of the vehicle and deciding the scope of the model, the model creation begins. As mentioned previously, while most real systems exhibit continuous behavior, L2 models are discrete; the main challenge in creating a model is creating a discrete representation of the system useful for diagnosis. Given that most of the EO-1 telemetry observations used by the model are already discrete, creating a discrete representation was straightforward.

The EO-1 model was created using L2's graphical model creation tool, called Stanley. It contains the three main subsystems described above: the ALI, the Hyperion, and the WARP.

The Hyperion model is shown in Figure 2 below.

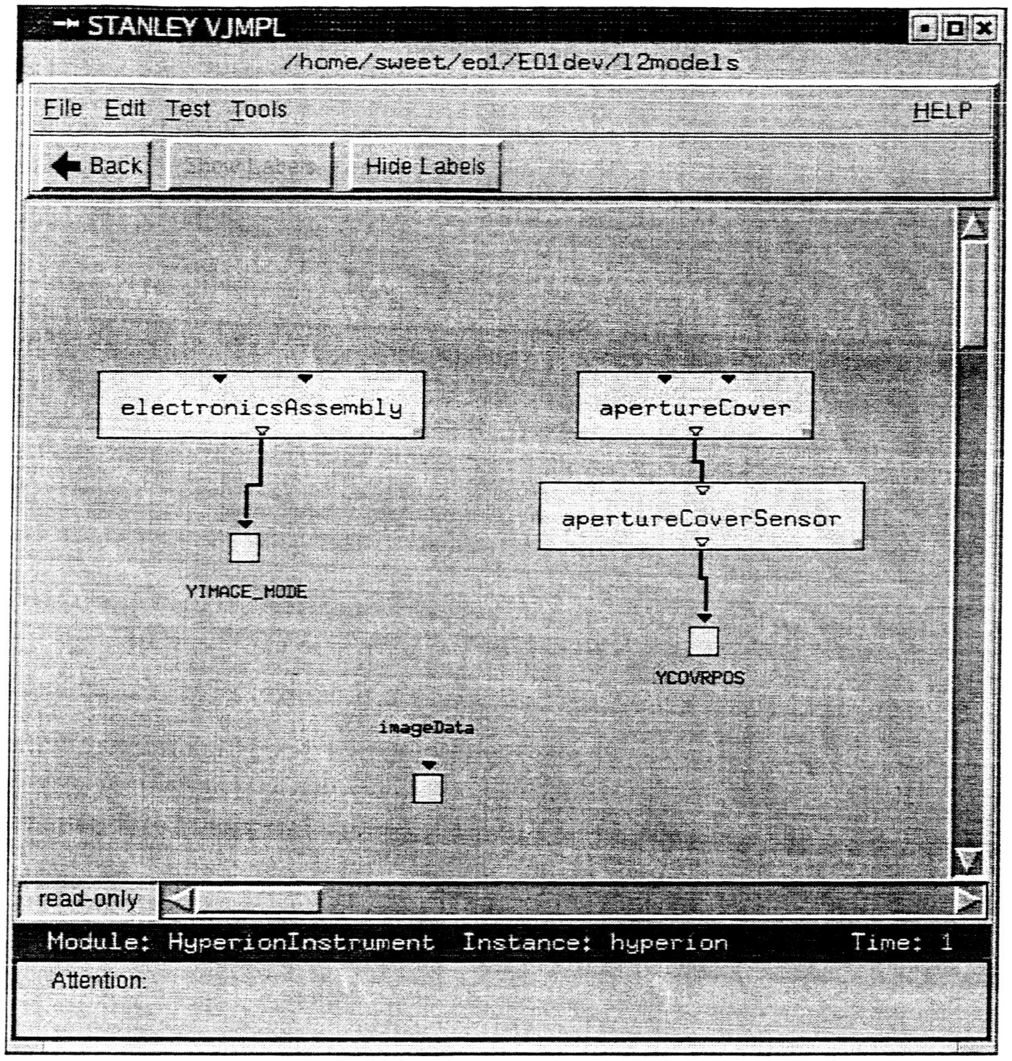


**Figure 2: L2 model of the Hyperion**

There are three components modeled in the Hyperion; the main aperture cover, which opens to image the earth, the aperture cover sensor, which measures the aperture cover's position, and the electronics assembly, containing the imaging electronics. The *imageData* variable represents what type of image is being taken. It is set based on the modes of the *electronicsAssembly* and the *apertureCover*: NO_IMAGE if the *electronicsAssembly* is disabled, DARK_IMAGE is the electronics are enabled but the *apertureCover* is closed, and EARTH_IMAGE if the aperture cover is open.

The ALI model is shown in Figure 3:

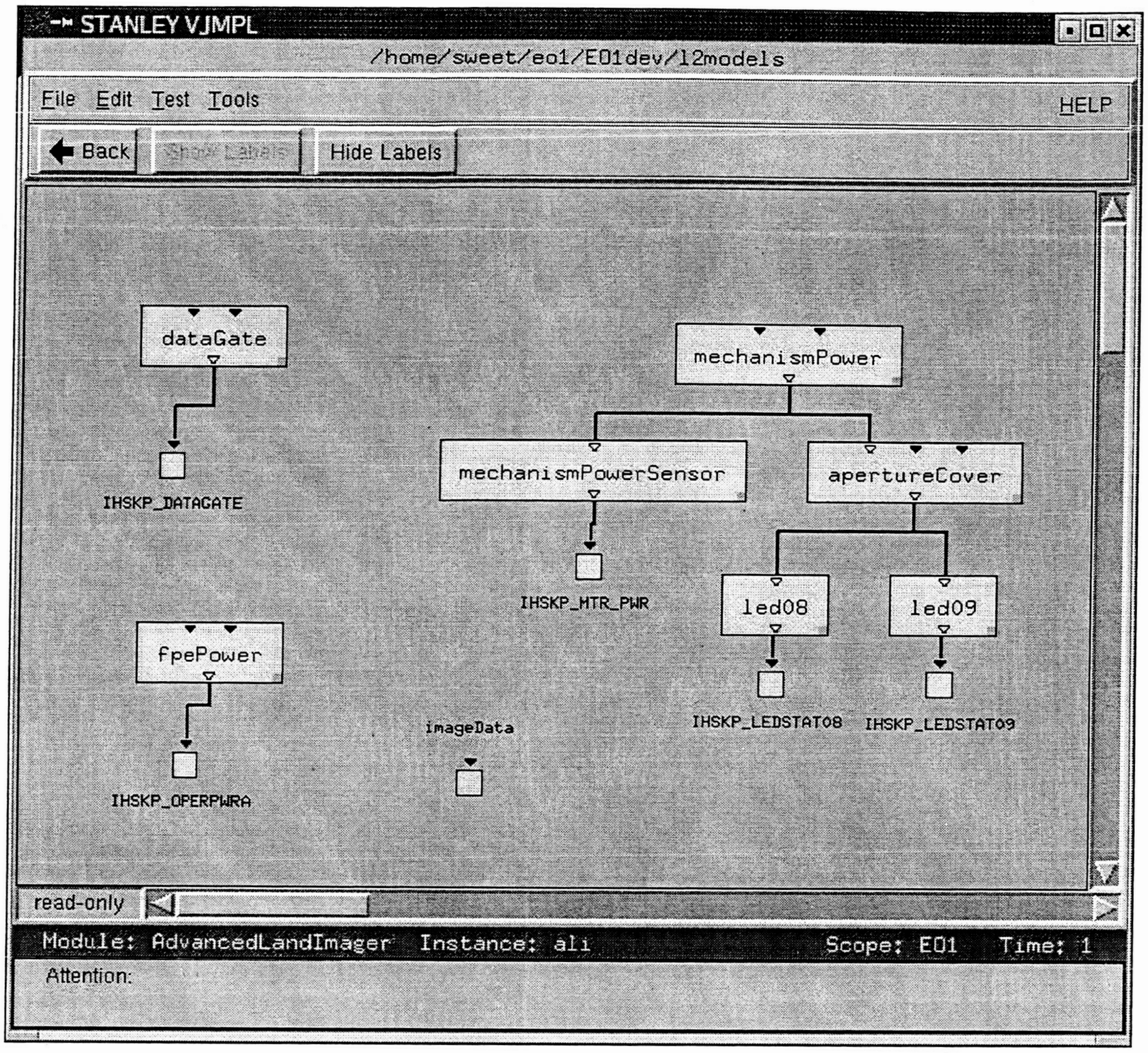


**Figure 3: L2 model of the ALI**

There are five components contained in the model of the ALI. First, the *fpePower* represents the focal plane electronics power, which must be enabled in order to take an image. The *dataGate* status indicates if an image is currently being acquired. The *apertureCover* acts as the lens cap of the instrument; it is normally closed to protect the instrument and to take dark calibration images, and open when taking images of the earth. The *mechanismPower* component supplies power to the aperture cover, allowing it to move. Again, the *imageData* variable represents what type of image is being taken. It is assigned according to the modes of the *fpePower*, *dataGate*, and *apertureCover*: if the *fpePower* or *dataGate* is disabled, it is set to NO_IMAGE; otherwise, if the aperture cover is closed, it is set to DARK_IMAGE, and if the *apertureCover* is open it is set to EARTH_IMAGE. Finally, three of the sensors are modeled: the *mechanismPowerSensor* which reports the state of the mechanism power, and two light-emitting diode (LED) indicators, which indicate the state of the *apertureCover*. The multiple sensors surrounding the *apertureCover* were explicitly modeled (and allowed to fail) because the semi-redundant information will allow L2 to find multiple hypotheses when a single fault occurs in the subsystem, one of the key features demonstrated in the experiment.

Finally, the WARP model is shown below:

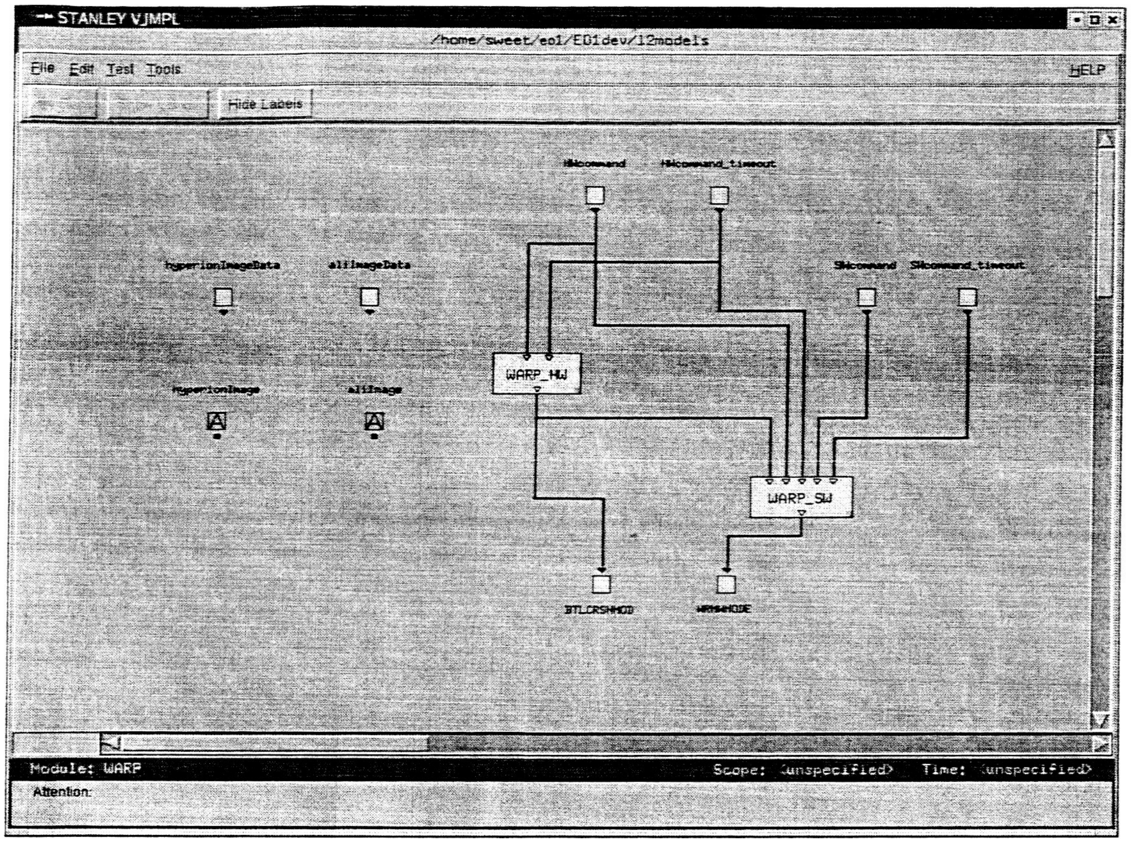


**Figure 4: L2 model of the WARP**

The WARP is modeled as two components, the WARP_HW and the WARP_SW. The WARP_HW models the device's hardware modes, and the WARP_SW models the software modes. As evidenced by the many connections between the two components, the hardware modes and software modes are closely related – not all combinations of modes are valid.

## D. Testing the Model

Models can be tested directly in the Stanley GUI model development environment. Execution of the models involves L2 in a stand-alone mode. A diagnostic scenario, consisting of a sequence of commands and telemetry observations, exercises the model within Stanley.

For testing the L2 model of the EO-1 satellite, the basic scenario is the same as the model's scope: the satellite's imaging timeline (data collection event). More specifically, the scenario is composed of the commands and telemetry observations sent to the Hyperion, ALI, and WARP. This sequence is briefly described below:

- Components set to image collection mode
- Dark calibration image taken
- ALI and Hyperion aperture covers opened
- Earth image taken
- ALI and Hyperion aperture covers closed
- Dark calibration image taken again
- Components set to standby mode

Seventeen Stanley scenarios were created in all, one representing the nominal data collection event, and one to test each of the fault modes in the L2 model. Each of the fault scenarios is based on the nominal scenario but with telemetry modified to inject the fault.

## V. Integration and Test on the EO-1 Test Beds

### A. Remote Testing on the Strings

Initial testing was conducted on a PowerPC computer. Two tests were defined: test C and D. Test C was designed to test the integration of the L2 and SCL software components. Test D had the addition of CASPER and the Science software. Once these tests were completed, the testing was moved to the "flight" like hardware test beds. These test beds consist of three PCs, two M-5s, and communication hardware—which all together was called a string. Goddard was the keeper and maintainer of three such strings. Later, JPL took String 2 into their possession while Ames continued to work on String 1 and 3 at Goddard.

A schematic is given below showing the different parts and how they connect.

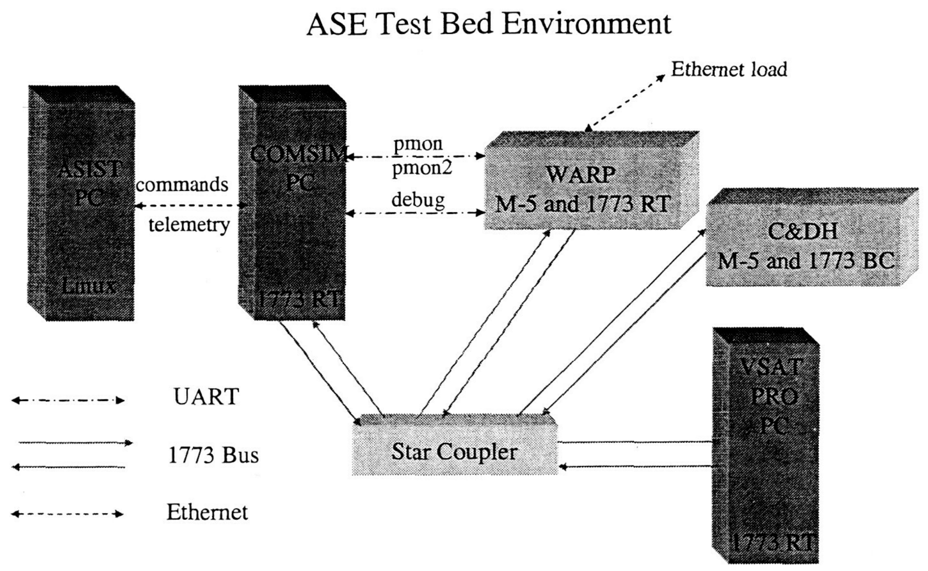


**Figure 5: Test Bed Hardware Configuration**

The Linux machine runs the ground station software, called ASIST. It handles the communication to and from the satellite. COMSIM simulates the communication hardware that would be receiving commands and telemetry. It's also is used to capture the serial output coming from the WARP so that it can be used for debugging purposes. COMSIM will also used to upload the software to the WARP via the Ethernet, which could also be done with another PC using TFTP. The Star Coupler is the 1773 network hub in which all the test bed hardware communicates through to each other. The WARP M-5 processor is where the experimental software resides. The C&DH is the 1773 Bus Controller (BC) and will talk to other 1773 Remote Terminals (RT), passing commands and receiving data. The VSAT Pro is a virtual satellite that simulates the satellite hardware, which consist of the satellite's attitude, instruments, WARP remote service node, and power.

### B. Test Automation

We developed the capability to perform automated testing. A scripting language called "Expect" was used in the automation of testing the L2 software on the M5 test bed at Goddard Space Flight Center (GSFC). "Expect" is an extension of the Tool Command Language (Tcl) and is used for automating user input to other applications. The name "Expect" comes from the idea of send/expect sequences popularized by uucp, kermit and other modem control programs.

"Expect" automated everything from starting up the software on the M5 test bed to running L2 to downloading the files and translating them into human-readable text. The only thing it was unable to do was to reboot the hardware; however, given time, this will eventually happen too as it already does this for the PowerPC test beds.

Just about every single scenario took about three hours to complete and required over 2,000 keystroke entries; that's no joke. With a three hour test, many things could go wrong. Automation of the test procedure eliminated the majority of things going wrong.

### C. CPU Metrics

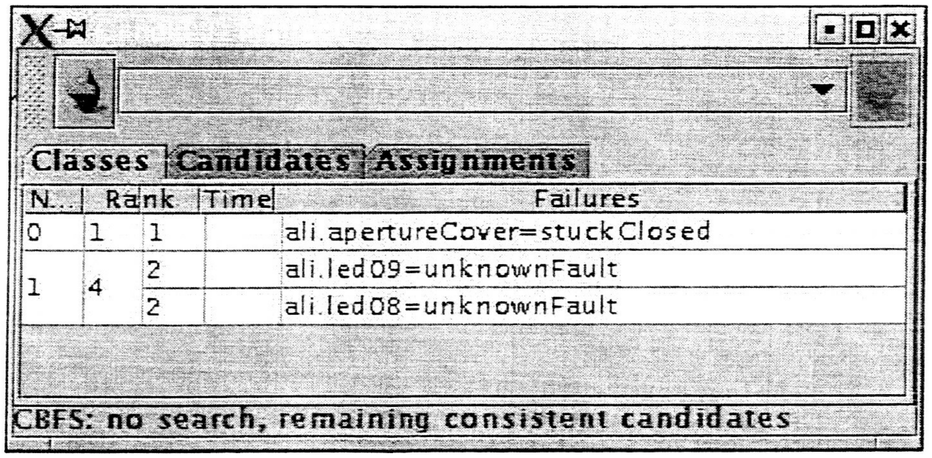
The VxWorks' "spy" utility was used to measure CPU utilization per task on the PowerPC only. This functionality is not supported on the M-5 strings, as these are veritable black boxes. A function was written to initialize the "spy" interrupt clock at 1000 ticks per second and to collect data at five second intervals. The task's output was directed to a file on the PowerPC target. Upon completion of the test, the report file was downloaded to the host computer and parsed with a Tcl script to extract the individual tasks' CPU usage data and save it to a Microsoft Excel file.

The data collected represented the number of 'ticks' each task executed, out of the 1000 possible samples per five seconds. This was then turned into a percentage of CPU time for each task.

## VI. Test Results

### A. L2 Unit Tests

In the L2 unit tests, the diagnosis from L2 is summarized graphically in the visualization tool Stanley, shown below. A screenshot from Stanley showing a diagnosis result is given below:



| N... | Rank | Time | Failures |
|---|---|---|---|
| 0 | 1 | 1 | ali.apertureCover=stuckClosed |
| 1 | 4 | 2 | ali.led09=unknownFault |
| | | 2 | ali.led08=unknownFault |

CBFS: no search, remaining consistent candidates

**Figure 6: Screenshot of L2 unit test diagnosis**

The relevant information in the table is that there are 2 candidates in the diagnosis of this fault. Each candidate is a possible explanation of the current observations. The first candidate contains a single fault, that the ALI's aperture cover is stuck closed. The second candidate contains two faults, that both of the LED sensors have failed. Each candidate is a possibility, according to the observations, but the single-fault candidate is more likely to have occurred (as indicated by the lower number in the "Rank" column. Here, the two LED sensors were measuring the position of the aperture cover. Hence, L2's diagnosis is that either the aperture cover is stuckClosed, or both of the sensors measuring the cover position have failed. This split of "component failed or sensors failed" is a common result when using L2.

The results for the all of the L2 unit tests are given in Table 1. The criterion for success is as follows:

1) The diagnosis contains the injected fault as a candidate

2) All other candidates in the diagnosis are also possible given the commands and observations

As we see from the table, the L2 unit tests completed successfully for all scenarios.

**Table 1: L2 unit test results**

| Scenario ID | Final Diagnosis Candidate(s) and Component Fault(s) | Correct? |
|---|---|---|
| eo1Nominal | none | Yes |
| eo1DualNominal | none | Yes |
| eo1FS01_AliDataGateFailedDisabled | ali.dataGate=failedDisabled<br>ali.dataGate=unknownFault | Yes |
| eo1FS02_AliDataGateFailedEnabled | ali.dataGate=failedEnabled<br>ali.dataGate=unknownFault | Yes |
| eo1FS03_AliMechanismPowerFailedDisabled | ali.mechanismPower=failedDisabled | Yes |
| eo1FS04_AliMechanismPowerFailedEnabled | ali.mechanismPowerSensor=unknownFault<br>ali.mechanismPower=failedEnabled | Yes |
| eo1FS05_AliMechanismPowerSensorFailed | ali.mechanismPowerSensor=unknownFault | Yes |
| eo1FS06_AliApertureCoverFailedClosed | ali.apertureCover=failedClosed<br>ali.led08=unknownFault<br>ali.led09=unknownFault | Yes |
| eo1FS07_AliApertureCoverFailedOpen | ali.apertureCover=failedOpen<br>ali.led08=unknownFault<br>ali.led09=unknownFault | Yes |
| eo1FS08_AliApertureCoverFailedIntermediate | ali.apertureCover=failedIntermediate<br>ali.led08=unknownFault | Yes |
| eo1FS09_AliLed09Failed | ali.led09=unknownFault | Yes |
| eo1FS10_AliLed08Failed | ali.led08=unknownFault | Yes |
| eo1FS20_HyperionApertureCoverFailedOpen | hyperion.apertureCover=failedOpen<br>hyperion.apertureCoverSensor=unknownFault | Yes |
| eo1FS21_HyperionApertureCoverFailedClosed | hyperion.apertureCover=failedClosed<br>hyperion.apertureCoverSensor=unknownFault | Yes |
| eo1FS23_HyperionElectronicsError | hyperion.electronicsAssembly=error<br>hyperion.electronicsAssembly=unknownFault | Yes |
| eo1FS24_HyperionApertureCoverSensorFailed | hyperion.apertureCover=failedOpen<br>hyperion.apertureCoverSensor=unknownFault | Yes |
| eo1FS35_WarpFailedToRecord | warp.software=unknownFault | Yes |

## B. Integrated Tests

The diagnosis output of the integrated tests on the ASE test bed environment is stored in text files. The same output will be used onboard the EO-1, and the text files will be compressed and downlinked to ground for analysis. The results of the integrated tests are given below in Table 2.

**Table 2: Integrated test results**

| Scenario ID | Final Diagnosis Candidate(s) and Component Fault(s) | Correct? |
|---|---|---|
| eo1Nominal | none | Yes |

| | | |
|---|---|---|
| eo1DualNominal | none | Yes |
| eo1FS01_AliDataGateFailedDisabled | none | No |
| eo1FS02_AliDataGateFailedEnabled | ali.dataGate=failedEnabled<br>ali.dataGate=unknownFault | Yes |
| eo1FS03_AliMechanismPowerFailedDisabled | ali.mechanismPower=failedDisabled<br>ali.mechanismPowerSensor=unknownFault | Yes |
| eo1FS04_AliMechanismPowerFailedEnabled | ali.mechanismPowerSensor=unknownFault<br>ali.mechanismPower=failedEnabled | Yes |
| eo1FS05_AliMechanismPowerSensorFailed | ali.mechanismPowerSensor=unknownFault | Yes |
| eo1FS06_AliApertureCoverFailedClosed | ali.apertureCover=failedClosed<br>ali.led08=unknownFault<br>ali.apertureCover=failedIntermediate | Yes |
| eo1FS07_AliApertureCoverFailedOpen | ali.apertureCover=failedOpen<br>ali.led08=unknownFault<br>ali.led09=unknownFault | Yes |
| eo1FS08_AliApertureCoverFailedIntermediate | ali.apertureCover=failedIntermediate<br>ali.led08=unknownFault | Yes |
| eo1FS09_AliLed09Failed | ali.led09=unknownFault | Yes |
| eo1FS10_AliLed08Failed | ali.led08=unknownFault | Yes |
| eo1FS20_HyperionApertureCoverFailedOpen | hyperion.apertureCover=failedOpen<br>hyperion.apertureCoverSensor=unknownFault | Yes |
| eo1FS21_HyperionApertureCoverFailedClosed | hyperion.apertureCover=failedClosed<br>hyperion.apertureCoverSensor=unknownFault | Yes |
| eo1FS23_HyperionElectronicsError | hyperion.electronicsAssembly=error<br>hyperion.electronicsAssembly=unknownFault | Yes |
| eo1FS24_HyperionApertureCoverSensorFailed | hyperion.apertureCover=failedOpen<br>hyperion.apertureCoverSensor=unknownFault | Yes |
| eo1FS35_WarpFailedToRecord | warp.software=unknownFault | Yes |

In the integrated test, 16 out of 17 scenarios completed successfully. The reason that FS01 failed the test is due to timing latencies of the actual system. The ALI data gate is commanded enabled, but commanded disabled again before the "enabled" telemetry was received. Therefore, there is no difference to L2 whether the commands succeeded or the component failed. L2 assumes no faults exist until evidence to the contrary is received; in this case that assumption results in a missed diagnosis for FS01.

Some other minor differences in the results also exist. For FS06, we have a different double-fault candidate. This candidate is as likely as the one found in the L2 unit test. Because of CPU restrictions, L2 is restricted from exhausting all possible candidates. Here, it simply found this double-fault candidate first. If L2 was not limited by CPU, it would have returned both double-fault candidates (for a total of 3 candidates) in both the unit test and the integrated test.

## VII. Conclusion

Over the past year, the project has gone from initiation to deployment on-board a spacecraft. Models of the satellite were developed from scratch and diagnostic scenarios validated on a series of test beds of increasing fidelity. A new Real-Time Interface and transient modeling methodology was employed to enable the software to run on a real-world system. We learned about the satellite, about operations procedures, and how to coax delicate hardware and firmware systems into a working state. The L2 software has been uploaded to the satellite and tests are about to begin.

Much work lies ahead. A future paper will report on results of the on-board validation tests. Further important work remains on implementing recovery once a diagnosis is made. The models could be extended significantly, and performance improvements can be made. We believe that this work will significantly contribute to the maturation of model-based diagnosis and improve the chances for adoption of this helpful technology for many missions and applications.

## References

1. J. Kurien and P. Nayak. "Back to the future for consistency-based trajectory tracking". *Proceedings of the 7th National Conference on Artificial Intelligence* (AAAI'2000), 2000.
2. N. Muscettola, P. Nayak, B. Pell and B. Williams, "Remote Agent: To Boldly Go Where No AI System Has Gone Before". *Artificial Intelligence,* Vol 100, Best of IJCAI 97.
3. S. Chien, R. Sherwood, D. Tran, R. Castano, et al., "Autonomous Science on the EO-1 Mission". *Proceedings of the Seventh International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Nara, Japan, 2003.

## Biographies

**Sandra Hayden** is Principal Investigator and Project Manager of the Livingstone on EO-1 (LEO-1) Infusion Experiment, and Group Manager for QSS Group Inc. Founded the QSS Software Process Improvement Network (SPIN) and lead CMMi pre-appraisals of selected QSS tasks. Architected and developed Integrated Vehicle Health Management (IVHM) for reusable launch vehicles (RLVs), under the Space Launch Initiative (SLI) program. Ames Project Manager, under the Next Generation Launch Technology program, for the PITEX project (Propulsion IVHM Technology Experiment for X-vehicles), an IVHM application for the X-34 RLV Main Propulsion System (MPS). Her interests include model-based diagnosis and flight-critical systems. Prior experience is on software engineering for large, mission and safety-critical Ada software systems such as the Canadian Automated Air Traffic System (CAATS) and a 1553 System Data Bus for a submarine. Ms Hayden received an MS in Computer Science from Simon Fraser University, Vancouver, Canada.

**Adam Sweet** is a research engineer in the Computational Sciences Division at NASA Ames Research Center, under contract to QSS Group Inc. He graduated with an MS in Mechanical Engineering from UC Berkeley in 1999, and has since worked at Ames modeling and simulating physical systems. His focus has been in robotics, hybrid system simulation, and model-based diagnosis.

**Scott Christa** is a systems engineer at NASA Ames Research Center under contract to AerospaceComputing, Inc. He graduated with a Bachelor of Science in Aeronautics from San Jose State University and holds an Airline Transport Pilot (ATP) and flight instructor certificates. Although he doesn't fly for a living, he specializes in programming embedded system using anything from high-level languages down to assembler code.